Amendments to the Claims

Please amend Claims 1, 3, 4, 9, and 10. Please add new Claims 11-19. The Claim Listing below will replace all prior versions of the claims in the application:

Claim Listing

1. (Currently Amended) A computer implemented method of analyzing multi-threaded programs, comprising:

suspending a first thread that requests a synchronization object that could result in a deadlock if acquired~~, the deadlock being evidenced by if another thread previously held the synchronization object while acquiring another synchronization object~~;

receiving a request from a second thread to acquire the synchronization object while the first thread is suspended;

allowing the second thread to acquire the synchronization object; and

causing ~~awakening~~ the first thread to awake in response to an event message to potentially produce a deadlock~~.~~;

wherein the method of suspending the first thread is in response to the first thread's request for the synchronization object that could result in a deadlock if acquired so evidenced by existence if any thread currently holding but not having released the synchronization object while acquiring another synchronization object.

2. (Original) The method of claim 1, further comprising checking whether the first and second thread are deadlocked by the first thread waiting to acquire a synchronization object that the second thread holds and the second thread waiting to acquire a synchronization object that the first thread holds.

3. (Currently Amended) The method of claim 1, wherein the first thread is suspended for a predetermined time~~, meaning that the first thread awakens after the predetermined time expires.~~

4. (Currently Amended) The method of claim 3, wherein the first thread is also suspended on an event.~~, meaning that the event awakens the first thread.~~

5. (Original) The method of claim 4, wherein the second thread sends the event that awakens the first thread.

6. (Original) The method of claim 1, wherein the first and second threads can hold a plurality of synchronization objects at a time.

7. (Original) The method of claim 1, wherein only one thread can hold the synchronization object at a time.

8. (Original) The method of claim 1, wherein only the first and second threads can release synchronization objects that each holds.

9. (Currently Amended) A computer program product for analyzing multi-threaded programs, comprising:

a computer useable medium having a computer readable program, wherein the computer readable program when executed on a computer causes the computer to:

~~computer code that suspends~~ request to a first thread ~~that requests~~ requesting a synchronization object that could result in a deadlock if acquired~~, the deadlock being evidenced by if another thread previously held the synchronization object while acquiring another synchronization object~~;

~~computer code that determines if another thread previously held the synchronization object while acquiring another synchronization object;~~

determine evidence of such resulting deadlock;

suspend the first thread;

receive a request from a second thread to acquire the synchronization object while the first thread is suspended;

~~computer code that suspends the first thread if another thread previously held the synchronization object while acquiring another synchronization object;~~

~~computer code that receives a request from a second thread to acquire the synchronization object while the first thread is suspended;~~

~~computer code that~~ allows the second thread to acquire the synchronization object; and

~~computer code that~~ cause ~~awakens~~ the first thread to awake in response to an event message to potentially produce a deadlock; ~~and a computer readable medium that stores the computer codes.~~

wherein the computer readable program suspend the first thread is in response to the first thread's request for the synchronization object that could result in a deadlock if acquired so evidenced by existence if any thread currently holding but not having released the synchronization object while acquiring another synchronization object.

10. (Currently Amended) The computer program product of claim 9, wherein the computer ~~readable~~ useable medium is ~~selected from the group consisting~~ any of a CD-ROM, floppy disk, tape, flash memory, system memory, and a hard drive~~, and data signal embodied in a carrier wave~~.

11. (New) The computer program product of claim 9, wherein the computer readable program checks whether the first and second thread are deadlocked by the first thread waiting to acquire a synchronization object that the second thread holds and the second thread waiting to acquire a synchronization object that the first thread holds.

12. (New) The computer program product of claim 9, wherein the first thread is suspended for a predetermined time.

13. (New) The computer program product of claim 12, wherein the first thread is also suspended on an event.

14. (New) The computer program product of claim 13, wherein the second thread sends the event that awakens the first thread.

15. (New) The computer program product of claim 9, wherein the first and second threads can hold a plurality of synchronization objects at a time.

16. (New) The computer program product of claim 9, wherein only one thread can hold the synchronization object at a time.

17. (New) The computer program product of claim 9, wherein only the first and second threads can release synchronization objects that each holds.

18. (New) The computer program product of claim 9, further comprising causing the first thread to awake in response to an expiration of time.

19. (New) The method of claim 1, further comprising causing the first thread to awake in response to an expiration of time.